Control de calidad de las especificaciones de software escritas en lenguaje natural(*)

Núria Castell, Olga Slavkova, Toni Tuells
Universitat Politècnica de Catalunya
castell/slavkova/atuells@lsi.upc.es
Yannick Toussaint
Laboratorio ARAMIIHS MATRA-Espace, CNRS,
toussain@irit.fr

Palabras clave: Adquisición del conocimiento. Procesamiento del lenguaje natural.

Resumen: *La complejidad creciente de los sistemas informáticos, ligada a una evolución muy rápida del hardware, ha originado en la Ingeniería del Software dos problemas importantes: la calidad del software desarrollado y la comunicación entre los participantes de un mismo proyecto, debido a las dificultades que plantea el uso de lenguaje natural como vehículo de la información técnica. El proyecto LESD (Linguistic Engineering for Software Development) relaciona la Ingeniería Lingüística con la Ingeniería del Software. La finalidad de LESD es el desarrollo de herramientas informáticas que permitan: a) crear una interpretación conceptual de las especificaciones funcionales o preliminares de software espacial en inglés; b) la evaluación, usando técnicas de Inteligencia Artificial, de la trazabilidad, completitud, consistencia, verificabilidad y modificabilidad como factores principales de la calidad de dichas especificaciones; y c) el control de la calidad de las especificaciones basado en la medición cuantitativa del software.*

1. Introducción

La realización de proyectos de software cada vez más complejos implica, muy a menudo, la colaboración de varios equipos, razón por la cual la Ingeniería del Software se enfrenta a dos problemas importantes: la fiabilidad del software desarrollado y las dificultades de comunicación entre los participantes de un mismo proyecto.

La Ingeniería del Software nace de la voluntad de racionalizar el desarrollo del software y de los sistemas informáticos. Cubre todos los aspectos de la informática, interesándose tanto por los aspectos de gestión de un proyecto como por la escritura de código o la definición de controles. La complejidad creciente de los sistemas informáticos, ligada a una evolución muy rápida del hardware, ha contribuido a justificar esta disciplina. Las inversiones tanto económicas como en recursos humanos son también factores determinantes.

Se han desarrollado trabajos muy importantes sobre los lenguajes de programación y sobre los métodos de diseño, por ejemplo OOD y HOOD [HOOD, 91]. Los lenguajes formales y los métodos de especificación muestran que la metodología se introduce en los proyectos en una fase cada vez más temprana. Efectivamente es muy importante poder detectar los errores cuanto antes durante el desarrollo de un proyecto. La detección demasiado tardía puede implicar importantes cambios metodológicos y un coste suplementario. Tomando como referencia el coste de corrección de un error en la fase de especificación, en [Pressman, 87] se estima que el coste de corrección de un error de especificación es de 1.5 a 6 veces más elevado cuando se detecta en la fase de desarrollo y de 60 a 100 veces superior si se detecta durante el mantenimiento.

El lenguaje natural es el medio de comunicación más usado a lo largo de todo el desarrollo y la explotación de un proyecto informático. La facilidad aparente que cada cual posee para usar el lenguaje puede generar problemas de comprensión ligados a la ambigüedad de las frases o a la incoherencia de los textos en los que el lenguaje natural se usa como vehículo de la información técnica. Nosotros nos interesamos, en particular, por las dificultades que plantea el uso del lenguaje natural en la fase de especificación.

Los trabajos realizados sobre el lenguaje, tanto lingüísticos y terminológicos como informáticos, tienen cada vez más aplicación a grandes universos técnicos tales como el espacio, las telecomunicaciones, la ingeniería nuclear... Los trabajos que relacionan la Ingeniería Lingüística con la Ingeniería del Software son muy escasos. Nosotros consideramos que el tratamiento automático del lenguaje natural puede aportar una ayuda esencial, tanto al nivel del tratamiento de las especificaciones, como al del mantenimiento, la reusabilidad y la explotación de un proyecto informático [Toussaint, 93].

Las normas de Ingeniería del Software definen las etapas de desarrollo del software y proponen factores de calidad tales como la trazabilidad (capacidad de guardar la traza entre la expresión de una necesidad y su realización informática), la coherencia o la modificabilidad (capacidad de modificar fácilmente una especificación). Los lenguajes formales no integran estos factores o bien integran alguno de ellos. Por ejemplo, la coherencia se suele integrar y se traduce por demostración de teoremas mientras, que la modificabilidad no suele tenerse en cuenta y la trazabilidad es considerada sólo en algunos lenguajes formales. Además existen otras características que contribuyen a la calidad de las especificaciones y que no son accesibles por técnicas matemáticas sino por la experiencia del diseñador informático.

El problema principal del uso de diferentes formalismos en las diversas etapas de desarrollo del software es el de encontrar la respuesta a la pregunta: ¿esta especificación es realmente la expresión de nuestra necesidad? El uso del lenguaje natural en la fase de especificación permite obtener más fácilmente un consenso entre los diferentes participantes del proyecto y por tanto posibilita la respuesta positiva a la pregunta anterior.

Este artículo está estructurado de la siguiente manera. En los apartados 2, 3 y 4 presentamos el interés del lenguaje natural en la fase de especificación, algunas normas de redacción y la descripción de varios sistemas ya desarrollados. El proyecto LESD se presenta en el apartado 5. Se describe en el apartado 6 el tratamiento, en este proyecto, de las especificaciones en lenguaje natural y el proceso de construcción de la base de requerimientos, que contiene la descripción conceptual de todos los requerimientos de las especificaciones. En los apartados 7 y 8 se proponen técnicas de inteligencia artificial para evaluar dos de los factores de calidad de especificaciones que el proyecto contempla: trazabilidad y completitud. En el último apartado se describe el esquema del modelo jerárquico de control de la calidad basado en la medición cuantitativa del nivel de calidad alcanzado por los factores.

2. El LN como herramienta de especificación

¿Qué lugar ocupa el lenguaje natural? La respuesta viene dada, en parte, por la pregunta anterior sobre la adecuación entre la especificación y las necesidades. Pese a la gran diversidad de lenguajes y de sistemas de ayuda a la especificación, existen numerosas situaciones en las cuales es difícil empezar un trabajo de diseño sin recurrir a la redacción de textos que den una primera definición de los problemas a resolver y de las restricciones a respetar. El lenguaje natural permite aunar mejor las necesidades del cliente y las soluciones propuestas por el ingeniero. Además, el trabajo de especificación propiamente dicho, en la medida en que requiere todos los recursos mentales del ingeniero, encuentra una expresión más directa en el lenguaje natural.

En las primeras fases de desarrollo, las especificaciones de software para sistemas complejos (espaciales, aeronáuticos, nucleares...) son documentos voluminosos, redactados frecuentemente en lenguaje natural. Una vez creados estos documentos, se enriquecen progresivamente en las sucesivas fases del ciclo de vida industrial. Incluso después de su formalización, la redacción original en lenguaje natural puede usarse en fases posteriores, durante el funcionamiento y mantenimiento del sistema informático desarrollado; en particular para verificar la adecuación con las opciones del diseñador o con las necesidades del cliente. Por tanto los ingenieros trabajan de forma casi continua con los documentos originales.

Así pues, el lenguaje natural tiene un papel importante en un proyecto de Ingeniería del Software y la mayoría de los lenguajes formales lo utilizan de forma simplificada para la redacción de comentarios. Lenguaje natural y lenguajes formales están abocados a cohabitar, pudiéndose argumentar largamente sobre las capacidades y las limitaciones de cada uno de ambos métodos de expresión.

3. Las normas de redacción

Para guiar la redacción de los documentos, existen normas que definen las restricciones lingüísticas que las especificaciones deben satisfacer. Hay dos grandes categorías de normas: unas inciden en el uso del lenguaje natural en general (por ejemplo, [IEEE, 84] y [AECMA, 89]) mientras que las otras se basan en restricciones terminológicas asociadas a un dominio particular (por ejemplo las normas de la ESA-European Space Agency).

Tanto unas como otras limitan el uso del lenguaje natural mediante un conjunto de restricciones que limita las diversas irregularidades (polisemias, paráfrasis, ambigüedades, imprecisiones...) que aparecen en la interpretación del lenguaje natural y que se traducen, al nivel de redacción, en tres grandes recomendaciones: uso de frases simples (estructuras sintácticas sencillas), uso de vocabulario restringido (tanto al nivel técnico como al coloquial) y eliminación de toda expresión vaga.

Aunque las normas definen restricciones muy precisas desde el punto de vista lingüístico, la realidad nos decepciona: a menudo las normas no se respetan y las consecuencias que esta falta de rigor provoca en el resultado final son difíciles de detectar a posteriori. No hay que olvidar que el tamaño de estos documentos no permite, incluso a un lector atento, detectar las variaciones en el uso de los términos ni controlar todos los significados de una frase con objeto de saber si otro lector puede interpretarla de manera diferente.

Además de las restricciones lingüísticas, las normas también incluyen restricciones de Ingeniería del Software referentes a los factores de calidad de las especificaciones: consistencia, completitud, trazabilidad, modificabilidad y verificabilidad.

4. Tratamiento de las especificaciones en LN

Desde el punto de vista del tratamiento del lenguaje natural para la especificación de software, existen dos enfoques diferentes, pero no excluyentes. Un enfoque se limita a las características superficiales de los textos en lenguaje natural (palabras, grupos de palabras) y su objetivo es gestionar un conjunto de requerimientos. El otro enfoque se interesa en el contenido informacional, la semántica.

El primer enfoque se basa en el desarrollo de entornos, generalmente a partir de sistemas de gestión de bases de datos, que ayudan al ingeniero en la tarea de archivo de las especificaciones. Estos entornos permiten insertar automáticamente la tabla de trazabilidad al final de los documentos. Sin embargo las relaciones asociadas a un requerimiento se han creado 'manualmente' por los ingenieros en el momento de la introducción del requerimiento. Como ejemplo de este enfoque podemos citar los sistemas ARTS [Dorfman, 84] o SWIFT.

El segundo enfoque intenta abordar el uso de especificaciones en lenguaje natural de acuerdo con los trabajos realizados sobre tratamiento automático del lenguaje natural y en particular sobre la utilización de sus estructuras lógico-semánticas. Este enfoque implica la necesidad de manejar representaciones semánticas. Construir la representación semántica de los requerimientos relativos a un dominio

determinado requiere no sólo conocimientos lingüísticos (sintácticos y semánticos) sobre la lengua en que están escritos los requerimientos sino también conocimiento especializado sobre el dominio en cuestión. Nosotros preferimos explorar en profundidad el lenguaje natural como herramienta de especificación y por tanto nos encuadramos en este segundo enfoque. A continuación comentamos brevemente algunos trabajos desarrollados siguiendo esta línea.

En primer lugar citamos el sistema SEC *(Simplified English Checker)*. Es un verificador automático de estilo en textos, desarrollado por la Boeing para ayudar a los redactores de documentos de mantenimiento en aeronáutica a ajustarse a los criterios de redacción de la norma AECMA PSC-85-16598. Puede consultarse una descripción de este sistema en [Wojcik, 90]. SEC ofrece al redactor un informe en el que se describe dónde y porqué el documento revisado se aparta de la norma. En 1990 el software desarrollado trataba correctamente el 80% del texto introducido y la compañía estimaba que tenía una ganancia de entre 12.000 a 16.000 horas de trabajo por año. Además de elevar la calidad de la redacción de los documentos, el uso de SEC tiene un efecto educativo muy positivo (el redactor aprende cuales son sus errores más frecuentes y los corrige; con ello disminuye el nivel de errores).

En segundo lugar citamos otro trabajo cuyo enfoque es más original, si bien nos parece menos realista. Se trata del sistema descrito en [Seki, 88] y cuyo objetivo es la traducción de especificaciones en lenguaje natural a un lenguaje formal. El lenguaje formal de especificación es un lenguaje algebraico que consta de una gramática de contexto libre y un conjunto de axiomas que definen la semántica del lenguaje. Este planteamiento nos parece muy limitado por diversas razones: el analizador usado es muy reducido y no trata, entre otros, el problema de la cuantificación, lo cual no permite la verificación de la coherencia a nivel del lenguaje formal; la traducción automática no permite introducir en la expresión formal aquellas informaciones que están implícitas en las especificaciones en lenguaje natural y que son esenciales. En realidad nos encontramos con la redacción de especificaciones en un 'lenguaje natural' tan reducido que casi se convierte en escribir especificaciones algebraicas mediante frases 'naturales'.

En tercer lugar citamos los trabajos de ([Carver, 89], [Cordes, 88] y [Cordes, 89]). Su método de evaluación de la calidad de las especificaciones en LN consta de dos fases: la primera es un análisis del texto para obtener un entorno de especificación formalizado y la segunda consiste en analizar la coherencia, completitud y trazabilidad basándose en el entorno construido en la fase anterior. Como resultado el sistema ofrece un informe del análisis realizado. Para crear el entorno formalizado se analiza el texto frase a frase (análisis prácticamente sintáctico) y luego se construye una base de hechos codificada en Prolog. Sobre esta base de hechos se aplican los diversos algoritmos definidos para realizar los controles de calidad mencionados previamente. Este trabajo es más la definición de una metodología que el desarrollo de un sistema real.

5. El Proyecto LESD

El proyecto LESD (Linguistic Engineering for Software Development) [Borillo, 91] nace en el centro ARAMIIHS de Toulouse (Francia), creado mediante acuerdo entre el CNRS y la empresa MATRA MARCONI SPACE. En el proyecto participan investigadores del IRIT (Institut de Recerche en Informatique de Toulouse), la Universidad de Ciencias Paul Sabatier, la Universidad de Letras Le Mirail, la empresa MATRA y algunos investigadores de la Universitat Politècnica de Catalunya, éstos desde hace dos años a través de una acción integrada hispano-francesa.

La finalidad de LESD [Toussaint, 91] es el desarrollo de herramientas informáticas que permitan crear una interpretación conceptual de especificaciones funcionales o preliminares en inglés, evaluar los factores de calidad (mencionados en el apartado 3) mediante algoritmos de razonamiento sobre la representación conceptual, y ayudar a la explotación de estos documentos. A más largo plazo, el objetivo de LESD es el desarrollo de un sistema de ayuda a la redacción que permita detectar los problemas ligados al uso de una terminología no identificada y de formas sintácticas complejas o ambiguas. Las especificaciones manejadas por LESD son especificaciones de software aeroespacial.

LESD explora en 'profundidad' el lenguaje y por tanto integra componentes léxicos, sintácticos y semánticos. Además se ha introducido el conocimiento del dominio ya que es absolutamente imprescindible para la construcción de la interpretación conceptual y para el razonamiento ([Borillo, 92], [Toussaint, 92]).

6. Arquitectura de LESD

La arquitectura actual de LESD, descrita gráficamente en la **figura 1**, se divide en dos partes: la primera engloba el análisis sintáctico-semántico y de dominio de los requerimientos, obteniéndose la representación conceptual de los mismos (por tanto nos situamos en el marco de la Ingeniería Lingüística); la segunda entra en el campo de la Inteligencia Artificial y engloba los mecanismos de razonamiento sobre la representación de los requerimientos.

6.1. Análisis de los requerimientos

Los requerimientos que componen una especificación se analizan sucesivamente utilizando el analizador ALVEY [Briscoe, 87], analizador que ha sido adaptado a nuestro dominio. Tras el análisis semántico y funcional, el requerimiento se interpreta tomando en consideración la representación del dominio. La representación generada se integra a la base de requerimientos. Una descripción detallada del proceso puede verse en [Toussaint, 92].

Los analizadores no están completamente desarrollados, pero se ha llevado a cabo una tarea importante de definición e im-

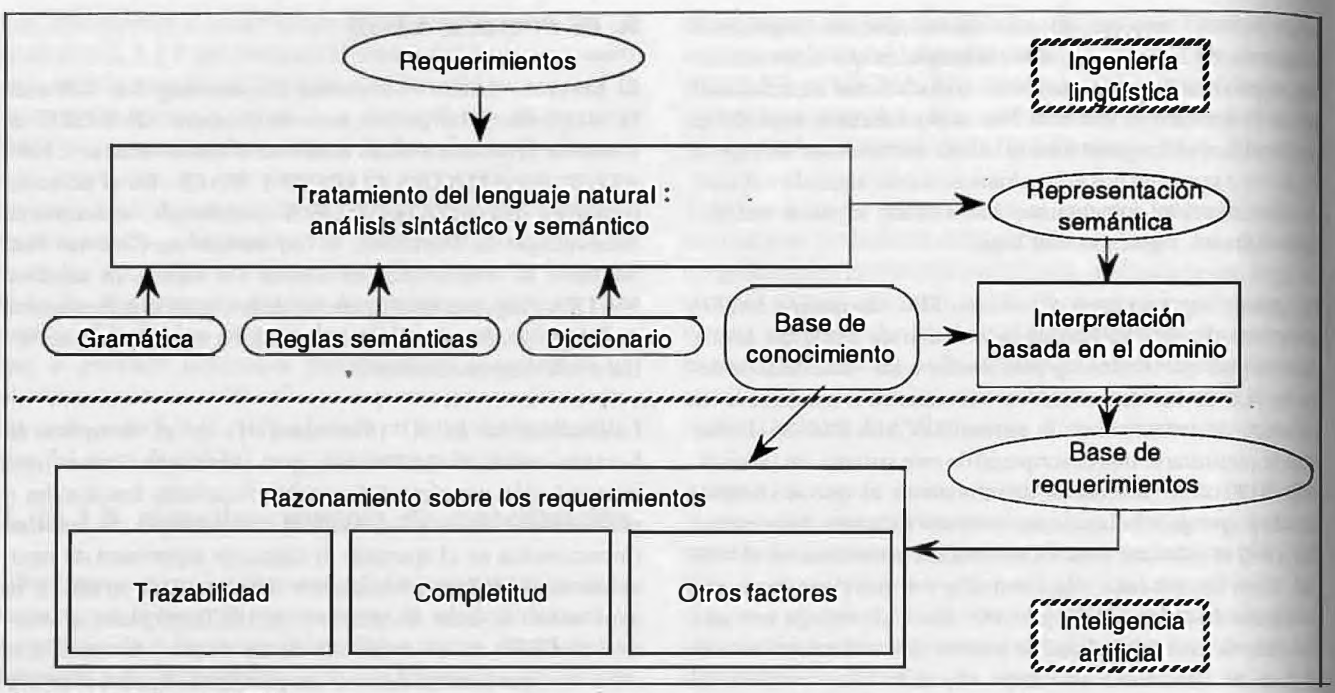

Figura 1: Arquitectura de LESD

plementación de la base de conocimiento y la de requerimientos. Se ha definido una tipología de los objetos y las actividades del dominio, así como las relaciones que hay entre ellos. Todo se ha implementado usando un formalismo basado en ***frames***.

6.2. Razonamiento sobre los requerimientos

El razonamiento sobre los requerimientos está concebido en varios módulos y cada uno de ellos corresponde a un factor de calidad. Se trata de mecanismos de razonamiento que utilizan la representación de los requerimientos contenida en la base de requerimientos. y la base de conocimiento general del dominio. Por ahora nos hemos ocupado de dos factores de calidad particularmente importantes en el diseño de software que presentamos en los apartados siguientes: trazabilidad y completitud.

7. La trazabilidad

A partir de las peticiones del cliente, el ingeniero escribe las especificaciones del sistema y de los subsistemas en un proceso 'top-down'. A esta etapa inicial de definición le siguen otras fases en las cuales se desarrollarán diversos módulos que serán integrados en un proceso 'bottom-up'. La trazabilidad expresa la conexión existente entre los requerimientos en los diferentes niveles. Durante el proceso 'top-down', es muy útil conocer el origen de los requerimientos. En el proceso 'bottom-up', la trazabilidad facilita los tests de control y permite identificar los requerimientos iniciales que no han sido satisfechos.

Existen diferentes enfoques en el diseño de herramientas de ayuda al ingeniero en las tareas de trazabilidad. Cuando las especificaciones están escritas en lenguaje formal, el control de la trazabilidad se basa en las propiedades algebraicas de las mismas. Otro enfoque consiste en usar algún sistema de base de datos y construir un entorno de desarrollo de especificaciones. En estos sistemas los enlaces de trazabilidad son creados manualmente por el ingeniero. Enfoques más recientes permiten el uso del lenguaje natural, siendo una posible propuesta la traducción automática del lenguaje natural a un lenguaje formal [Seki, 88].

7.1. La trazabilidad en LESD

En LESD nos hemos planteado la siguiente pregunta: ¿Cuáles son los criterios lingüísticos (semánticos y pragmáticos) involucrados en la creación de enlaces de trazabilidad entre requerimientos?

El control de la trazabilidad no consiste en una simple búsqueda de correspondencias entre todos los conceptos que aparecen en los requerimientos. Este control implica el uso de técnicas de inferencia en una representación muy estructurada del conocimiento del universo de discurso. Uno de los primeros objetivos de LESD ha sido la definición de un conjunto de relaciones para estructurar el léxico y los conceptos del dominio. Hemos establecido relaciones taxonómicas (la relación 'is-a'), meronímicas (descomposición de un objeto en sus componentes), temporales (especialmente entre actividades), de caracterización (por ejemplo 'status' caracteriza 'system') y de funcionalidad temática ('agente', 'objeto'...).

Actualmente no existe una definición formal de la trazabilidad que permita implementar un algoritmo que, de forma automática, genere los enlaces de trazabilidad. Por tanto hemos optado por un enfoque interactivo en el cual el ingeniero plantea una consulta precisando un conjunto de conceptos ligados entre sí mediante relaciones. La respuesta del sistema

consiste en una lista de requerimientos cuya representación conceptual posee esos conceptos y relaciones. El algoritmo desarrollado para analizar las consultas y calcular las respuestas está basado en la noción de **tipo**. Por tanto los requerimientos seleccionados son aquéllos cuyos conceptos son del tipo del concepto dado en la consulta. Esto permite clasificar los requerimientos del más específico al más general en función de la derivación (mediante la relación 'is-a') entre los conceptos presentes en la consulta y los conceptos presentes en cada requerimiento. El cálculo de las respuestas supone la activación de un mecanismo de inferencias sobre la base de conocimiento.

Las consultas pueden ser muy simples, por ejemplo *"Obtener todos los requerimientos relacionados con un tipo de objeto"* o bien tener un importante nivel de complejidad, obligando a la realización de inferencias en la estructura y el contenido de la base de conocimiento. Por ejemplo, *"Obtener todos los requerimientos que expresan la monitorización de datos"* incluirá los siguientes requerimientos en la respuesta:
- req.1: *The systems shall monitor the data of the space vehicle*
- req.2: *The ioi-gs shall receive the data of the space vehicle*

El sistema puede relacionar estos requerimientos gracias a la taxonomía de objetos y las relaciones existentes entre ellos. Por ejemplo: 'ioi-gs' es una subclase de 'system', la actividad 'receive' es una subactividad de 'monitoring' y 'data of space vehicle' es una subclase de 'data'. Además el sistema detecta que los conceptos presentes en el req. 2 son más específicos que los que aparecen en el req.1. Por tanto, satisfacer el req.2 supone satisfacer el req.1 sólo parcialmente.

8. La completitud

¿Qué se entiende por completitud cuando hablamos de especificaciones de software escritas en lenguaje natural? La palabra completitud se utiliza en tantas ramas de la ciencia que a veces no se sabe si tiene siempre el mismo significado o tiene varios diferentes. Nosotros creemos que sí que se utiliza la palabra 'completitud' con significados diferentes. A continuación repasamos brevemente algunos casos concretos, prestando especial atención a su relación con el procesamiento del lenguaje natural. Para una descripción detallada ver [Tuells, 94].

8.1. Completitud deductiva

Esta idea de la completitud es la que se aplica a cualquier sistema formal y es, por ejemplo, la que se utiliza en los teoremas de incompletitud de Gödel. Una posible definición sería: *"un sistema formal es completo si para cualquier proposición A del sistema, o bien A o ¬A pueden ser deducidas'.*

8.2 Completitud expresiva

Completitud expresiva es la relacionada con la expresividad (o falta de expresividad) de lenguajes de programación, formalismos lingüísticos o sistemas de representación del conocimiento. Evidentemente, esta falta de expresividad depende de aquello que se quiera expresar; es decir, depende de la habilidad para representar las clases de conocimiento que son necesarias en el dominio.

8.3 Completitud estructural

Hablamos de completitud estructural en relación a la ausencia o no de una subestructura dentro de una estructura más general. Hay bastantes ejemplos de estos tipos de estructuras: algunos de los casos más conocidos en procesamiento del lenguaje natural son los *scripts* de Schank [Schank, 77] y los diálogos orientados a tareas [Grosz, 86]. Vale la pena destacar que estas estructuras son más o menos completas en relación a aquello que se pretende modelizar (por ejemplo, se acostumbra a utilizar un modelo de usuario).

Respecto a la Ingeniería del Software, se han propuesto soluciones similares a las anteriormente citadas para controlar la completitud de un conjunto de especificaciones. Por ejemplo, en el proyecto **'The Requirements Apprentice'** [Reubenstein, 91] se utiliza el concepto de **'cliché'** (que en realidad es un ***frame***) asociado a un dominio concreto. En cualquier cliché hay unos valores por defecto asociados a ciertos *'slots'* y otros que se deben concretar. Un requerimiento no se considera completo hasta que no se hayan especificado todos los valores de los slots que están obligados a tener algún valor.

8.4. Completitud conceptual

Está relacionada con la comunicación entre personas diferentes, y por tanto con el lenguaje natural, que es el vehículo de comunicación más habitual entre personas. Desde este punto de vista, un **acto de comunicación**, que para simplificar entenderemos como la emisión de un mensaje por parte de un emisor y su recepción por parte de un receptor, **es completo si el receptor comprende aquello que el emisor desea comunicar**. En relación a las especificaciones software escritas en lenguaje natural, diremos que son completas si permiten que el ingeniero de software comprenda cual es la funcionalidad del sistema. El problema principal de estas especificaciones, y que provoca que valga la pena estudiarlas, es que **la persona que las escribe no es la misma que la que las ha de leer**.

8.5. Control de especificaciones incompletas en LESD

Para este control es necesaria la construcción de una jerarquía de acciones-subacciones lo suficientemente representativa como para asegurar que ésta debe aparecer en cualquier conjunto de especificaciones asociadas a un dominio espacial. Por ejemplo, en nuestra base de conocimiento la actividad 'monitorizar' se descompone en tres subactividades: 'recibir', 'analizar' y 'visualizar'.

Consideremos el siguiente ejemplo real de requerimiento:

"During the launch phase, the IOI-GS shall analyse and display the status of the space vehicle"

Si después de analizar este requerimiento el sistema observa que no hay referencia alguna a la actividad de 'monitorizar', avisará al ingeniero de una posible falta de requerimiento relativo a esta actividad. De forma similar, si después de analizar los requerimientos se observa que uno de ellos hace referencia a la actividad 'monitorizar', pero no hay ninguno que haga referencia a sus descendientes en la jerarquía, entonces el sistema puede sugerir al ingeniero que las especificaciones son incompletas.

Esta propuesta utiliza mecanismos de razonamiento sobre la base de conocimiento. Es necesario que dicha base sea fiable y precisa. El ingeniero deberá controlar la coherencia de las inferencias realizadas por el sistema y validar qué requerimientos son necesarios. Además el ingeniero debe ser siempre el árbitro de la completitud de las especificaciones; ésto es debido a que tanto nuestra posible solución como la que se propone en [Reubenstein, 91] intentan 'capturar' la completidud conceptual de las especificaciones via **completitud estructural** de una estructura (la jerarquía de acciones-subacciones en nuestro caso). Esta aproximación supone un tratamiento parcial del problema de la completitud sobre el que estamos trabajando.

9. Medición de la calidad

La fase de especificación tiene clara influencia sobre la calidad del software; por lo tanto es necesario controlar rigurosamente el nivel de calidad de las especificaciones evaluando cada uno de sus factores. La evaluación de los factores de calidad de las especificaciones considerados en LESD supone el desarrollo de algoritmos de razonamiento sobre la representación conceptual de las especificaciones funcionales o preliminares en lenguaje natural. Los algoritmos desarrollados permiten definir formalmente la descomposición de los factores en criterios cuantitativos de su calidad, más fáciles de evaluar que los factores.

La evaluación del nivel de calidad obtenido por cada uno de los criterios se realiza por métricas de software. Las métricas son procedimientos que definen la evaluación del nivel de la calidad de dichos criterios en función de sus componentes que pueden ser medidos directamente (basándose en el algoritmo correspondiente). Las mediciones cuantitativas de la calidad de los componentes son funciones que tienen como parámetros dichos componentes y devuelven un simple número que se interpreta por la correspondiente métrica como el nivel de calidad alcanzado por el componente. El formalismo en la definición de los factores, los criterios, los componentes de la calidad y todas las relaciones entre ellos nos garantiza el control objetivo y riguroso de la calidad de las especificaciones.

En general el modelo del control de calidad del software tiene estructura jerárquica [Slavkova_92]. Establecemos cinco niveles en el modelo que vamos a usar para controlar la calidad de las especificaciones:

I nivel: refleja el propósito general del sistema de control de la calidad. En nuestro caso sería la calidad de las especificaciones.

II nivel: se define por los factores. En nuestro caso serían la trazabilidad, la consistencia, la completitud, la verificabilidad y la modificabilidad.

III nivel: se define por los criterios de calidad. Por ejemplo, el factor trazabilidad se descompone en dos criterios:
- trazabilidad de todas las especificaciones;
- enlace entre los requerimientos en diferentes niveles de integración de módulos.

IV nivel: corresponde a las métricas del software. Para los dos criterios de la trazabilidad las métricas correspondientes pueden definirse de la siguiente forma:
- la relación entre el número de especificaciones trazables y el número total de especificaciones;
- la relación entre los requerimientos enlazados de diferentes niveles de integración de módulos y el número total de enlaces entre estos niveles según los requisitos del usuario.

V nivel: corresponde a las medidas (mediciones cuantitativas) de los componentes de la calidad. Para calcular las métricas señaladas se necesitarían las medidas de los siguientes componentes de calidad:
- número de especificaciones trazables;
- número total de especificaciones;
- para los módulos de diferentes niveles de integración, calculo de:
  . número de requerimientos enlazados;
  . número de enlaces según los requisitos del usuario.

El modelo jerárquico de control de calidad del software permite flexibilidad en la actualización de cada uno de sus niveles. Por lo tanto una herramienta que lo implemente puede ser adaptada fácilmente a los cambios en las definiciones de los factores, criterios, métricas, medidas y sus relaciones. Hasta el momento está definido el algoritmo de evaluación del factor 'trazabilidad'; por lo tanto se ha podido definir su descomposición en criterios y las métricas que pueden evaluarlos en base a la medición cuantitativa de los componentes de la calidad [Slavkova, 93]. Después de definir formalmente los factores restantes, éstos se van a descomponer en niveles siguiendo la misma metodología. El resultado será un sistema que permitirá controlar la calidad de las especificaciones.

10. Conclusiones y trabajo futuro

El trabajo hasta ahora realizado en la primera fase del proyecto LESD ha consistido en el desarrollo de herramientas para el análisis de especificaciones escritas en lenguaje natural. Más concretamente, ha consistido en el desarrollo de los analizadores sintáctico y semántico de dichas especificaciones, el estudio del conocimiento necesario para la interpretación de las mismas, el diseño de un sistema de representación del conocimiento adecuado y la implementación de ciertos meca-

nismos de razonamiento para la evaluación de los factores de calidad de las especificaciones, siempre dentro de un dominio espacial.

Se han fijado cinco factores de calidad de las especificaciones (trazabilidad, completitud, consistencia, verificabilidad y modificabilidad), habiéndose creado ya las técnicas de evaluación de la trazabilidad, y actualmente se están desarrollando y estudiando las correspondientes a la completitud.

En un futuro inmediato pensamos estudiar el resto de factores de calidad. En función de los algoritmos de evaluación obtenidos, se construirá el sistema de control de calidad de las especificaciones basado en la medición cuantitativa del software.

11. Referencias

[AECMA, 89] Association Européene des Constructeurs de Matériel Aréonautique: *"A ECMA Simplified English, A Guide for the preparation of aircraft maintenance documentation in the international aerospace maintenance language"* (Diciembre 1989).

[Borillo, 91] Borillo M., Toussaint Y., Borillo A.: *"Motivations du project LESD"*. Conferencia Linguistic Engineering'91, Versailles, Francia (16-17 Enero 1991).

[Borillo, 92] Borillo M., Castell N., Latour D., Toussaint Y., Verdejo M.F.: *"Applying Linguistic Engineering to Software Engineering: The traceability problem"*. The European Conference on Artificial Intelligence (ECAI92), Viena, Austria (Agosto 1992).

[Briscoe, 87] Briscoe T., Grover C., Boguraev B., Carroll J.: *"The Alvey Natural language Tools Project Grammar: a Large Computational Grammar"*. Documentación de ALVEY, Universidad de Cambridge, Computer Laboratory, UK (1987).

[Castell, 94] Castell N., Slavkova O., Tuells T., Toussaint Y.: *"Quality Control of Software Specifications Written in Natural Language"*. Report LSI-94, Universitat Politècnica de Catalunya, Barcelona, España, (pendiente de publicación).

[Carver, 89] Carver D.L., Cordes D.W., Gautier N.: *"Object-Based Measurement in the Requirements Specification Phase"*.

[Cordes, 88] Cordes D.W., Carver D.L.: *"Knowledge Base Applications within Software Engineering: A Tool for Requirements Specification"*. First International Conference on Industrial & Engineering Applications of A.I. & Expert Systems, Tullahoma, USA, pp.266-280 (1-3 Junio 1988).

[Cordes, 89a] Cordes D.W., Carver D.L.: *"Evaluation Method for User Requirements Documents"*. Information and Software Technology, vol.31, no.4, pp. 181-188 (Mayo 1989).

[Cordes, 89b] Cordes D.W.: *"Improved Utilization of Requirements Document Information During System Specification"*, Proceedings of the 27th Annual Southeast Regional Conference, pp. 273-277, Coulter N.S. y Ellis E. eds., Atlanta, USA (5-7 Abril 198).

[Dorfman, 84] Dorfman M., Flynn R.F.: *"Arts - An Automated Requirements Traceability System"*. The Journal of Systems and Software, vol. 4, pp. 63-74 (1984).

[Gazdar, 85] Gazdar G., Klein E., Pullum G., Sag I.: *"Generalized Phrase Structure Grammar"*. Harvard University Press, Cambridge, UK (1985).

[Grosz, 86] Grosz B.: *"Attentions, Intentions, and the Structure of Discourse"*. Computational Linguistics, vol.12, n.3 (Julio-Setiembre 1986).

[HOOD, 91] Group H.O.T.: *"HOOD Reference Manual"*. HOOD (Julio 1991).

[IEEE, 84] *"IEEE Guide to Software Requirements Specifications"*. ANSI/IEEE Std 729-1983 (1983).

[Pressman, 87] Pressman R.S.: *"Software Engineering: A Practitioner's Approach"*. McGraw Hill, New York, USA (1987).

[Reubenstein, 91] Reubenstein H., Waters R.: *"The Requirements Apprentice: Automated Assistance for Requirements Acquisiton"*. IEEE Transactions on Software Engineering, vol. 17, n.3 (Marzo 1991).

[Schank, 77] Schank R.C., Abelson R.P.: *"Scripts, Plans, Goals and Understanding"*. Lawrence Erlbaum Associates Publishers, New Jersey, USA (1977).

[Seki, 88] Seki H., Nabika e., Matsumura T., Sugiyama Y., Fujii M., Torii K., Kasami T.: *"A Processing System for Program Specifications in a Natural Language"*. 21st. Annual Hawai International Conference on System Sciences. Vol.II: Software Track, pp. 754-763, ed. IEEE Comp. Soc. Press, Washington, USA (1988).

[Slavkova, 92] Slavkova O.: *"Métricas de software"*. Report LSI-92-6-T. Universitat Politècnica de Catalunya, Barcelona, España (1992).

[Slavkova, 93] Slavkova O.: *"Modelo para el control de calidad en LESD basado en la medición del software"*, Report LSI-93-26-R, Universitat Politècnicade Catalunya, Barcelona, España (1993).

[Toussaint, 91] Toussaint Y., Borillo M., Borillo A., Castell N., Latour D.: *"Applying Linguistic Engineering to Software Engineering"*. 26 Coloquio de Lingüística, Poznan, Polonia (18-21 Septiembre 1991)

[Toussaint, 92] Toussaint Y.: *"Méthodes Informatiques et Linguistiques pour l'aide a la Spécification de Logiciel"*. Tesis Doctoral. Universidad Paul Sabatier, Toulouse, Francia (1992).

[Toussaint, 93] Toussaint Y., Borillo M.: *"Natural Language in Software Engineering"*. En Encyclopedy of Software Engineering, John Wiley eds, (pendiente de publicación).

[Tuells, 94] Tuells T., Castell N:, *"The completeness problem in LESD"*. Report LSI-94, Universitat Politècnica de Catalunya, Barcelona, España.

[Wojcik, 90] Wojcik R.H., Hoard J.E., Holzhauser K.: *"The Boeing Simplified English Checker"*. Conférence sur l'Intelligence Artificielle dans l'Aéronautique, Toulouse, Francia (1990).

(*) Nota: Trabajo parcialmente financiado por la CICYT (TIC93-420) y por los gobiernos español y francés a través de una acción integrada hispano-francesa.